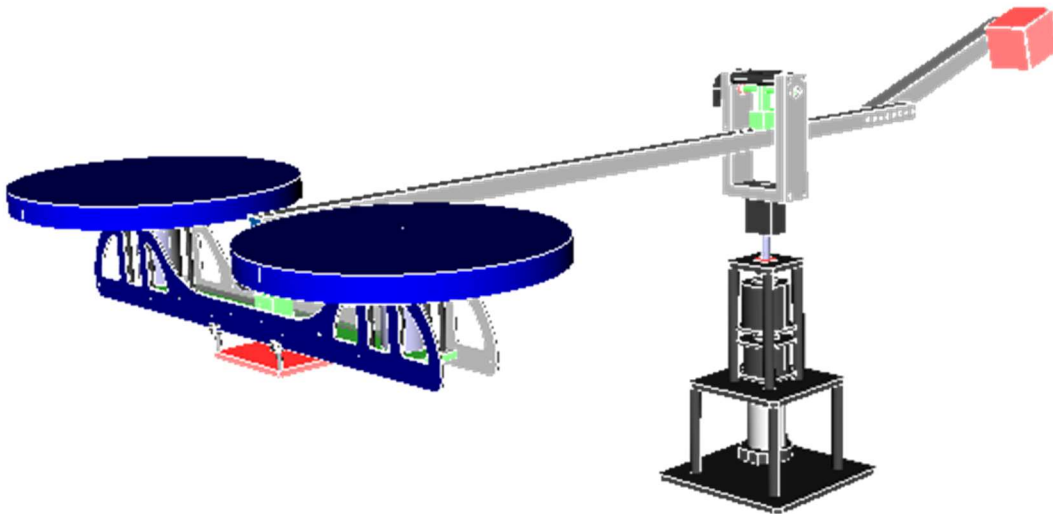

AER 715 AVIONICS AND SYSTEMS

Laboratory 4.2

Flight Control – Control System Design



Fall 2024

Rev. 1.10.1

Engineering Specialist: C. Chan

Instructor: Dr. G. Liu

TABLE OF CONTENTS

1.	Instructions	3
2.	Flight Control – Control System Design	4
2.1	Introduction	4
2.2	Purpose	4
2.3	Parameters of the 3-DOF Helicopter	5
2.4	Theory	6
	Proportional Gain	8
	Derivative Gain	8
	Integral Gain	8
2.5	Pre-Lab Assignment	10
3.	Lab Work	10
3.1	Part A: Designing a PID Controller in SISOTOOL	10
3.2	Part B: Simulation of Helicopter via a Virtual Reality (VR) Model	14
3.3	Part C: Post Lab	16

1. Instructions

- SAFETY FIRST – DO NOT PUT YOUR FINGERS OR ANY LOOSE ITEMS IN THE SERVOMOTOR GEARS.
- This lab is to be done **in groups of two (2)**.
- Download the lab manual, worksheet, and files from D2L and save them on the Desktop in a folder called LAB4.
- Read the instructions in the laboratory manual carefully and follow the specified procedures.
- Answer all questions in the provided worksheet.
- At the end of the lab, submit one lab worksheet along with the standard Ryerson Aerospace Assignment/Laboratory Cover Sheet. Each student must attend the laboratory and sign the Cover Sheet in order to receive a mark.

2. Flight Control – Control System Design

2.1 Introduction

In Lab 4.1 we have developed a mathematical model for the elevation and travel dynamics of the 3-DOF helicopter using analytical and experimental techniques. In this lab we will use the models to develop a combined elevation and travel control system with proportional-integral-derivative (PID) controllers.

We haven't yet formally covered the PID family of controllers, though we did design a proportional velocity controller in AER509 to control a servomotor. If you remember the proportional velocity controller, the velocity control gain operated on the output of the plant directly, and as such when we determined the closed-loop transfer function we found that there was a closed-loop pole added to the system. By maintaining the second order system, it allowed us to use second order system control design techniques to estimate the closed-loop performance.

2.2 Purpose

The objective of this lab is to design and simulate a control system to hold the helicopter at a prescribed elevation and command it to travel for a defined (yaw) angle of rotation. You will use MATLAB's SISOTOOL to design the control system and then incorporate your control design into Simulink.

Once you have adequately developed the controller, you will be able to simulate the helicopter and visualize its performance using a virtual model.

At the end of this laboratory, you should understand the following:

- How to design a PID controller for the 3-DOF helicopter.
- How to implement your controller in Simulink and evaluate the closed-loop performance.

2.3 Parameters of the 3-DOF Helicopter

The parameters of the 3-DoF helicopter are given in the following two tables:

Table 1: 3-DOF Weights and Measures

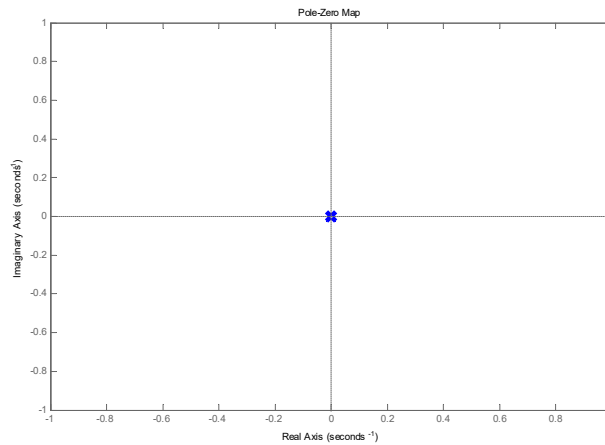
Symbol	MATLAB	Description	Unit	Value			
				Heli 1	Heli 2	Heli 3	Heli 4
M_h	Mh	Mass of Heli Body	[kg]	1.442	1.422	1.464	1.450
M_c	Mc	Mass of CW	[kg]	1.914	1.916	1.919	1.918
L_a	La	Distance from Pivot to Helicopter body centre	[in]	25.75			
L_b	Lb	Distance from Pivot to counterweight centre	[in]	18.125		18.5	
L_h	Lh	Distance from pitch axis to rotor center	[in]	6.985	6.932	6.995	6.933
J_e	Je	Moment of Inertia	[kg-m ²]	TBD	TBD	TBD	TBD
D_e	De	Viscous Damping	[N-m-s/rad]	TBD			
K_e	Ke	Spring Constant	[N-m/rad]	TBD			
F_t	Ft	Lift Force @ SLF	[N]	TBD	TBD	TBD	TBD

Table 2: Other Parameters and Limits of the 3-DOF Helicopter

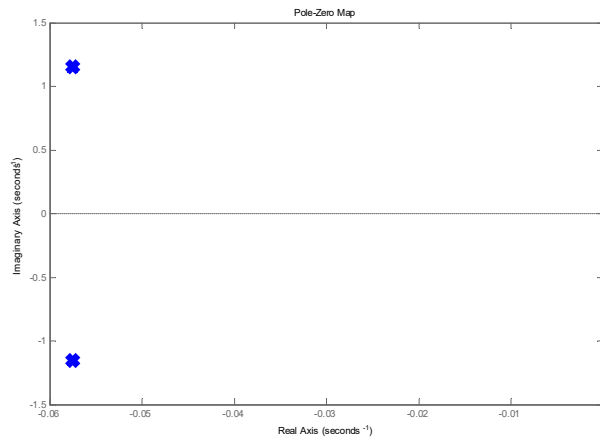
Symbol	MATLAB	Description	Unit	Value			
				Heli 1	Heli 2	Heli 3	Heli 4
K_f	Kf	Motor-Prop Force Constant	[N/V]	0.140			
K_{rt}	Krt	Motor-Prop Torque Constant	[N.m/V]	0.0036	0.0032	0.0038	0.0027
ϵ		Elevation Range	[Degrees]	[~-26 to ~30]			
ϵ_o		Elevation Start	[Degrees]	-25.75			
λ		Travel Range	[Degrees]	0 to 360			
g	g	Gravity constant	[m/s ²]	9.81			
	KE_CNT	Encoder Resolution	[counts/rev]	-4096			
	KE_RAD	Encoder Resolution	[rad/count]	1.5340E-2			
	K_CABLE	Amplifier Gain	[V/V]	3		5	

2.4 Theory

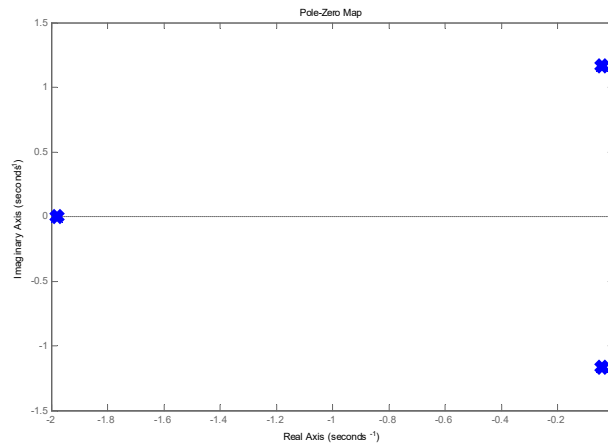
In Lab 4.1 you developed three distinct open-loop transfer functions for the elevation dynamics of the helicopter: 1) system without natural response, 2) second-order system generated from the system identification results of the actual step response (average of the three experimental trials), and 3) third-order (with the added motor dynamics) system generated from the system identification results as mentioned. Figure 1 illustrates the response of each of the three systems where the first plot shows a system with the presence of only the forced response (single pole at the origin); the second plot shows a system with the underdamped natural response (two poles on the left side of the imaginary axis without zeroes); and the last plot shows a third-order system with the added motor dynamic that is modeled as a first-order system. Now, you will design a PID (proportional-integral-derivative) controller for each model and simulate its results.



System without natural response



Second-order system (underdamped condition)



Third-order system (underdamped condition)

Figure 1 Poles marked for the three elevation transfer functions developed in Part 1

From the plots you should immediately see that the second and third systems are inherently stable as they have poles on the left side of the imaginary plane. The first system has two poles at $s=0$ and is unstable. Another thing you should notice is that the dominant complex poles in system two and three are very close to the imaginary axis, thus these systems can easily be destabilized. One of your primary goals in developing your controller is to move the closed-loop poles further into the left hand plane to improve the helicopter's performance.

Now let us review some theories on the PID control. A PID controller can be mathematically described as:

$$c(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{d}{dt} e(t) \quad \text{Equation 1}$$

where $c(t)$ is the PID controller output, $e(t)$ is the error term associated with the commanded input minus the actual system output, K_p is the proportional gain, K_i is the integral gain, and K_d is the derivative gain.

The purpose of the PID controller is to continuously generate the control signal that is sent to the plant in order to improve the closed-loop system performance. The controller is cascaded with the plant as shown in Figure 2 below.

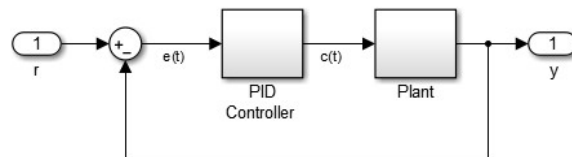


Figure 2 Classical feedback control scheme with PID controller.

Each gain component serves a specific purpose and can be used alone (by setting the other two gains to zero) or in conjunction with one or both the other terms. If we look inside the PID controller block, we will see how Equation 1 can be modeled in Simulink.

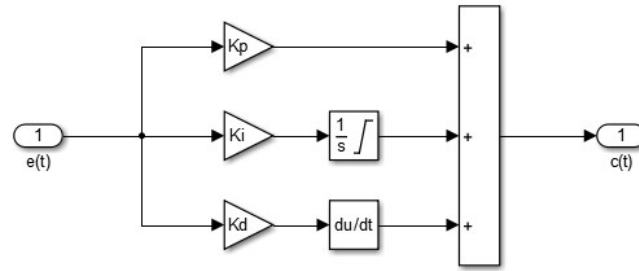


Figure 3 Parallel form of PID controller

We will look at each term specifically and examine what it aims to achieve.

Proportional Gain

The proportional gain K_p amplifies the error signal $e(t)$ directly. The proportional gain increases the actuating signal $c(t)$ to drive the error down to zero. One limitation of using only proportional control is that it normally cannot eliminate steady-state error. This is due to the simple fact that as the error becomes smaller and smaller, the actuating signal decreases proportionally and will eventually become insufficient to drive the plant. For example, imagine you are pushing a wooden block across the floor and your goal is to position the block some prescribed distance away. You begin to push the block forward with a force relative to your distance away from the block. As you steadily approach the final position you apply less force. At some point the force you are applying will be less than the frictional force acting on the block and you will be permanently stuck at a position short of your goal.

Derivative Gain

The derivative gain K_d amplifies the time derivative of the error signal $e(t)$, which is in fact a rate error. It helps damp the system response and is often used to help reduce overshoot. It is sometimes called anticipatory control, because it acts as if the controller is anticipating the approach of the zero error level. The derivative gain is effectively related to the damping in the system. The greater the gain, the greater the damping we add to the system.

The derivative gain cannot be used to eliminate steady-state error. It adds a zero in the closed-loop transfer function, making control design more difficult. It should be noted that if the error signal is noisy the derivative gain may amplify the noise.

Integral Gain

The integral gain K_i amplifies the time integral of the error signal $e(t)$. The benefit of the integral term is that it can help eliminate the steady state error. The integral term accumulates over time, increasing the actuating signal until the error reaches zero. Another benefit of the integral term is that if the error signal is noisy, the integral term acts like a low pass filter, thus smoothing the actuating signal.

There are several drawbacks with the integral term and they are listed as follows:

1. It responds relatively slowly to the error signal.
2. It adds a pole at $s = 0$, thus it can lead to system instability.
3. It can have a very large initial output at the moment a large error is produced.
4. Its cumulative growth (integral windup) could cause the actuation signal to saturate.

The last three items are usually of most concern. When adding integral term, the control system designer must ensure that the added pole will not destabilize the system within the operational bandwidth of the system. This can be achieved through analysis, simulation and testing. The large initial error in the output response due to a sudden large error can be mitigated by only activating the integral term within a specific error range. The cumulative growth problem can be relaxed using one of several integral anti-windup techniques available. The basic premise is to put limits on the output of the integral term so that it does not saturate the actuation signal or device.

2.5 Pre-Lab Assignment

This lab involves simulating the elevation dynamics of the 3-DOF helicopter. You will use both analytical and experimental means to determine the dynamic characteristics of the helicopters.

STEP	DESCRIPTION/TASK
1	Take the Laplace transform of Equation 1 to get the transfer function $\frac{c(s)}{e(s)}$ of a PID controller.
2	How many poles and zeroes does the PID controller have?

3. Lab Work

3.1 Part A: Designing a PID Controller in SISOTOOL

In this section we will design a PID controller for the second order system studied analytically in Lab 4.1. Our goal is to obtain a system response with an overshoot of no more than $\%OS = 5$, and a settling time of no longer than $T_s = 12$ seconds. We also want to have zero steady state error $\varepsilon_{ss} = 0$.

STEP	DESCRIPTION/TASK
1	Download the files for Lab 4 from D2L and put them in a Lab 4 folder on your desktop.
2	Start “ MATLAB ” and change the working directory to Lab4.
3	<p>Create the open-loop second order transfer function (e.g. G#_elev1) of the helicopter you used for Lab 4.1 in the Command window using the tf command and construct a closed-loop system with a unity feedback. Your open-loop transfer function should have the form:</p> $G\#_elev1(s) = \frac{K}{s^2}$ <p>and your closed-loop transfer function should have the form:</p> $T(s) = \frac{K}{s^2 + K}$ <p>Then, use the SISOTOOL in MATLAB to plot both the open-loop and closed-loop poles of G#_elev1 on the root locus diagram. To execute the SISOTOOL, use the following command: sisotool('rlocus', G_elev1)</p> <p>When you click on the Root Locus Editor for LoopTransfer_C tab, a figure with a plot of the root locus where the blue cross represents the open-loop pole and the pink squares represent the closed-loop poles will appear as shown in Figure 4.</p>

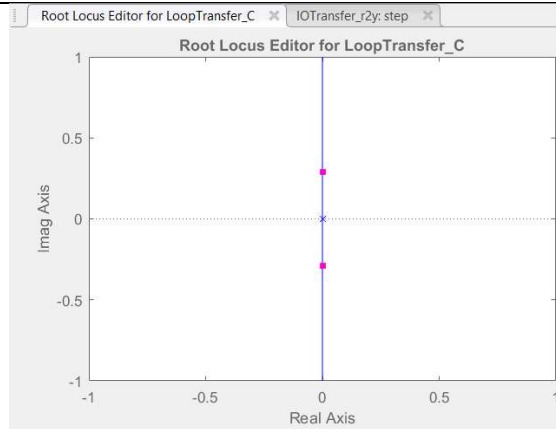


Figure 4 Root Locus plot of the elevation dynamics

First, let's adjust the scale of the real axis. Right click on the root locus plot and select **"Properties"**. Go to the **"Limits tab"** and set the **Real Axis Limits: -3 to 0** and the **Imaginary Axis Limits: -3 to 3**. We may need to make more adjustments later.

Now in the Root locus plot window, let us add our design criteria for the %OS and T_s . Right click on the plot and choose **Design Requirements >> New** and input **12** for the **Settling time**. Repeat this procedure and add the **%OS** design requirement as **5**. See Figure 5.

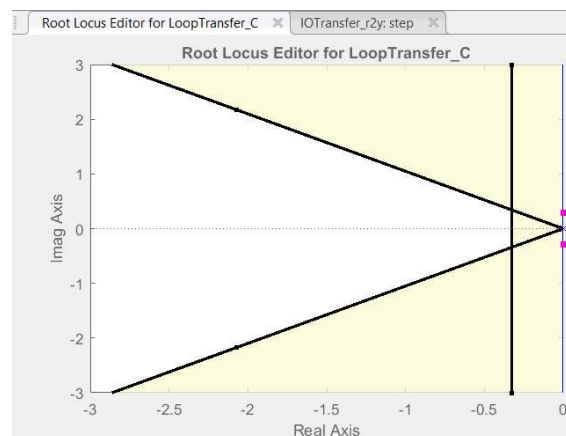


Figure 5 Root Locus with Design Requirements

A vertical line representing the settling time, and two angled lines representing the overshoot (or damping) will appear.

Note: These design criteria are also valid for higher order systems whose dominant poles are far right to the other poles. In such case, we approximate the system as second-order when conducting the root locus analysis.

Now, our goal is to have the closed-loop poles of our system be at the intersection points of our design criteria.

We know from the Pre-lab, that the PID controller consists of two zeroes (real or complex) and one pole at the origin as described in the following equation where z_{lag} and z_{lead} are the lag zero and lead zero respectively:

$$G(s) = \frac{K(s+z_{lag})(s+z_{lead})}{s} = \frac{K(z_{lag}z_{lead}+(z_{lag}+z_{lead})s+s^2)}{s} \quad \text{Equation 2}$$

Let us add the zeroes (z_{lag} and z_{lead}) and the lone pole at the origin (integrator) of the PID controller. Right click in the plot and select **Add Pole/Zero > Integrator**. Repeat and add a **complex zero**. Now the plot should look like the one depicted in Figure 6.

5

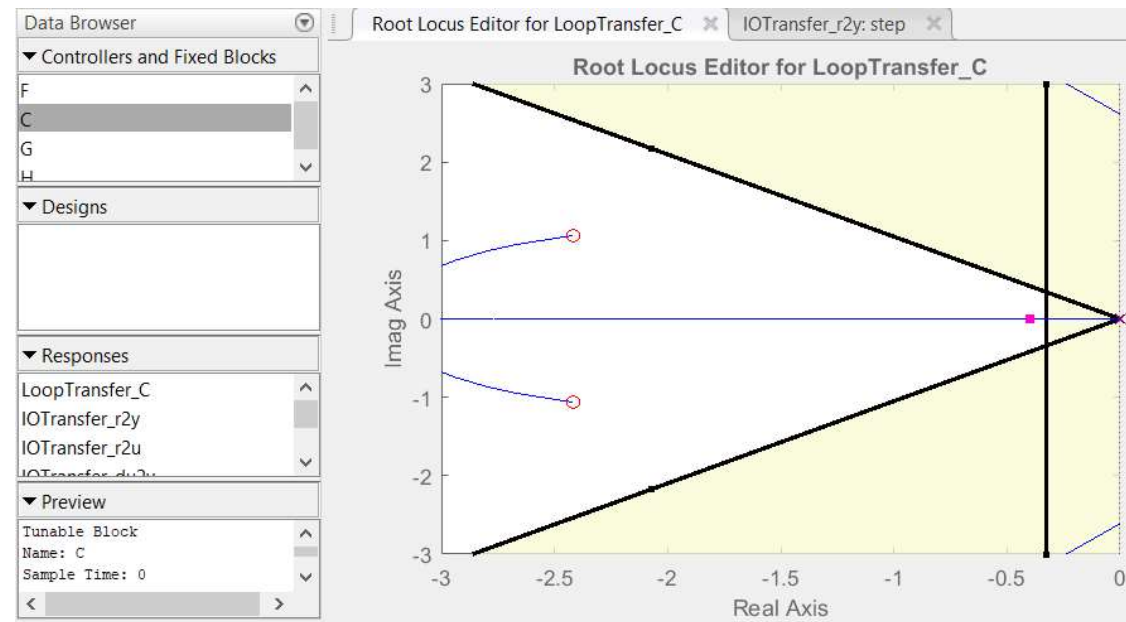


Figure 6 Root Locus with Added Integrator and Complex Zero

Obviously, if we want our closed-loop poles to pass through the intersection of our design requirements, we need to first adjust the position of the compensator zeroes. You can do this by clicking and dragging the compensator zeroes (pink circles). Drag the zeroes around until you see the paths of the root locus plot pass through the design intersection points. Once the lines pass through, grab the closed-loop pole (pink squares) and drag them to the intersection point. You should end up with something like the plot shown in Figure 7.

6

Next, click on the **IOTransfer_r2y: step** tab and look at the step response (Figure 8) to verify if we have achieved the %OS and the T_s as desired.

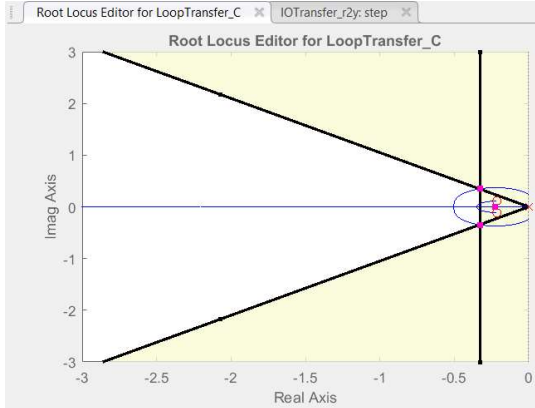


Figure 7 Closed-loop poles at the Design Requirements.

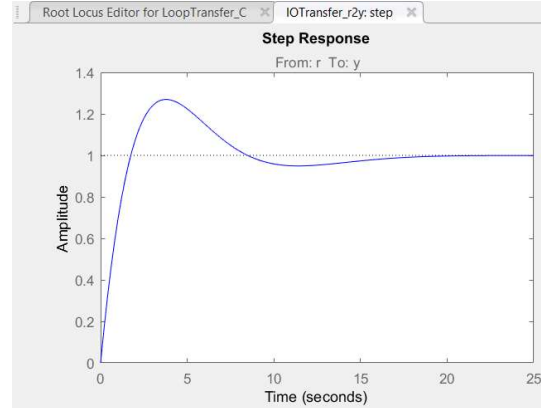


Figure 8 Step Response of closed-loop system showing overshoot and settling time labels.

Looking at Figure 8, **we see that we haven't quite reached our goals**. This should really come as no surprise since the root locus design criteria applies only to systems that resemble closely to second order systems. In order to achieve our desired system response, we will have to further fine-tune the zeroes of our PID compensator.

Now that we have developed a PID compensator in SISOTOOL, let us export the compensator and use it to simulate our system in Simulink.

In the **Controllers and Fixed Blocks** window, click on **C** (Compensator), and select **Export** from the top. Your designed compensator with the variable name **C** will be exported to the Workspace.

In order to determine the gain values of our PID controller, we first need to expand Equation 2 and convert it to the parallel format that we are used to:

$$G(s) = \frac{K(z_{lag} \cdot z_{lead} + (z_{lag} + z_{lead})s + s^2)}{s} = \frac{K_d \left(\frac{K_i}{K_d} + \frac{K_p}{K_d} s + s^2 \right)}{s} \quad \text{Equation 3}$$

7 where K_p , K_i , and K_d are the gains of the PID controller. When accomplish this in MATLAB, simply type the following in the Command Windows:

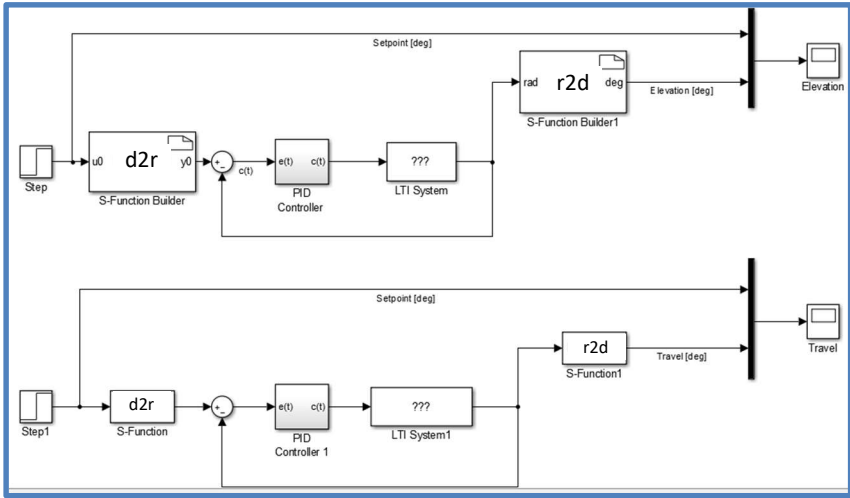
C_PID=pid(C)

Then extract each of the gains into their own variable as follows:

Kp=C_PID.Kp; Ki=C_PID.Ki; Kd=C_PID.Kd;

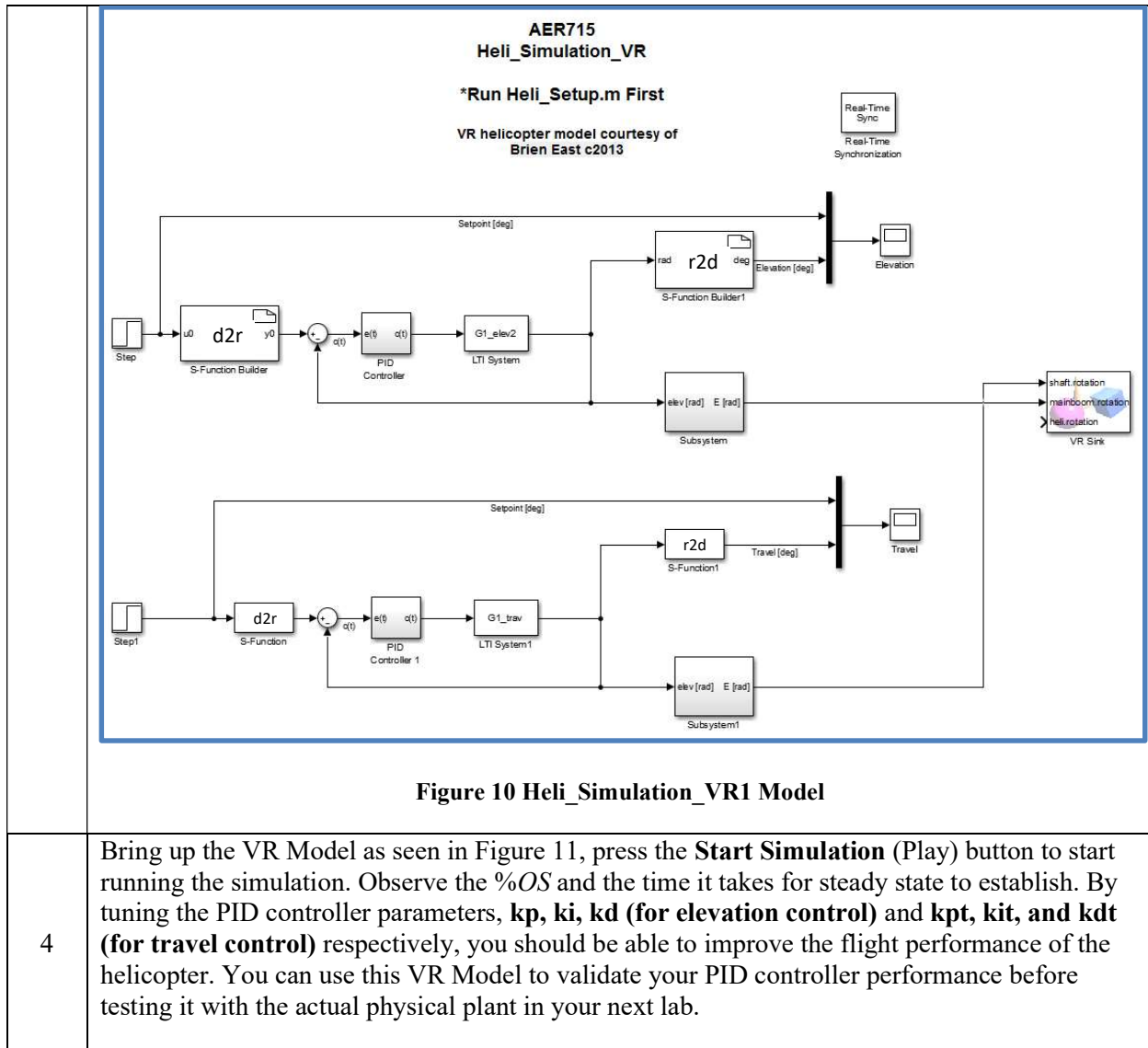
Now, go to your LAB4 files folder and make sure to load the **G#_trav** and **G#_trav_PID** files into the MATLAB workspace by copying them into your working directory and double clicking them

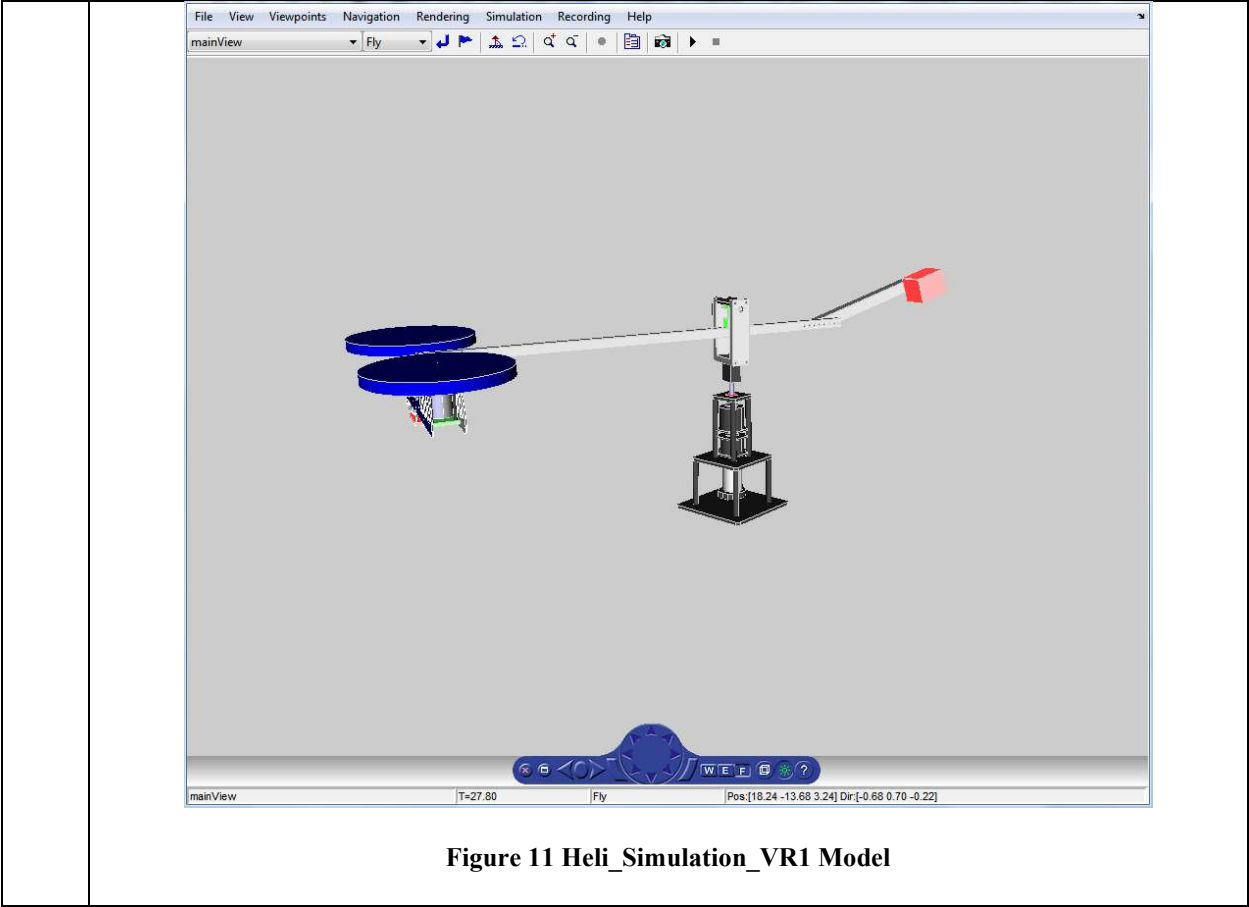
Note: The variables are case sensitive.

8	<p>Open the Simulink model “Heli_Simulation.slx” that you made in Lab 4.1; it should be similar to the one as shown in Figure 9. The model contains your transfer function and the gains for the PID controller.</p>  <p style="text-align: center;">Figure 9 Closed-Loop Control Model for 3DOF Helicopter</p> <p>Note: The G#_elev and G#_trav transfer functions must be in the MATLAB workspace for the LTI System blocks to recognize them, so make sure to open the files corresponding to your helicopter station and load them in before running.</p>
9	<p>Open the “Heli_Setup.m” script, then type “PID1” into the MATLAB command window (make sure the G#_trav_PID file is loaded), change the Kpt, Kit, and Kdt variables in the setup script corresponding to the ones you see in the transfer function.</p> <p>Run the “Heli_Setup.m” script file and then execute the simulation. Look at all of the scopes including the voltages being generated by the PID controller (PID controller block). The step response should be similar to what you had seen in SISOTOOL.</p> <p>Do this for PID1, PID2, and PID3.</p>

3.2 Part B: Simulation of Helicopter via a Virtual Reality (VR) Model

STEP	DESCRIPTION/TASK
1	Download the files for Lab 4 from D2L and put them in a Lab 4 folder on your desktop.
2	Start “ MATLAB ” NOT “MATLAB 2015b” (This VR Model does not work in the older version of MATLAB) and change the working directory to Lab4.
3	Open the Simulink model “ Heli_Simulation_VR1.slx ” as seen in Figure 10, with your calculated PID controller parameters from the previous steps (You may have to go into the blocks of the model to enter your own PID controller parameter values). Make sure you have run the script “ Heli_Setup.m ” and the transfer functions G#_elev and G#_trav have both been loaded in the MATLAB workspace.





3.3 Part C: Post Lab

STEP	DESCRIPTION/TASK
1	<p>In a new script, type the following:</p> <pre>%----- % AER 715 Introduction to Avionics and Systems % Lab 4 - "Lab Title" % Your Full Name(s) & SID(s) %----- % %% Introduction % Type your introduction in this section % %% Post Lab Exercises - % Put your exercises in this section % %% Conclusion % Write your lab conclusion for the WHOLE lab in this % section. % % % %</pre>

2	<p>In the SISOTOOL adjust the zeroes and/or gains to achieve a $T_s < 12$ seconds. Keep the following restrictions in mind: $\%OS < 5$ and Loop Gain < 100. Iterate your design until you achieve the Settling Time requirement.</p> <p>Complete this task for all three transfer functions you developed in Lab 3.</p>
3	<p>Create a new section called Question 1 using the %% command. Include a plot of the final root locus as in Figure 9 for each transfer function. Compare and discuss.</p>
4	<p>Create a new section called Question 2 using the %% command. Include a list of the PID gain values for each transfer function. Do this for both the elevation and the travel PID gains.</p>
5	<p>Create a new section called Question 3 using the %% command. Determine the complete closed-loop transfer function using the <i>feedback</i> command. Do this for both the elevation and the travel transfer functions.</p> <p>Note: There are three PID controllers for one travel transfer function</p>
6	<p>Create a new section called Question 4 using the %% command. Include a plot of the step responses of the closed loop systems. Include all elevation closed-loop transfer functions in one plot. Compare and discuss.</p> <p>Do the same for the travel system, but in a different plot. Discuss why the Kdt gain is so much higher than both the Kpt and Kit gains based on this plot.</p>
7	<p>Create a new section called Question 5 using the %% command. For the elevation systems, include a list of the estimated $\%OS$, T_s and T_r from the closed-loop transfer functions. Compare and discuss.</p> <p>Hint: There is a MATLAB function to get this data.</p>
8	<p>Create a new section called Question 6 using the %% command. Include plots of the individual gain voltages (one plot per PID controller). The PID controller values are saved in the variable <u>CV</u> in the Workspace (only elevation data is included in the variable). Compare and discuss.</p>
9	<p>Publish your script and submit along with the standard Ryerson Aerospace cover sheet. Keep a copy of your PID gains as we will use them in the next lab.</p>